

REMARKS

These remarks are set forth in response to the Office Action. As this amendment has been timely filed within the three-month statutory period, neither an extension of time nor a fee is required. In paragraph 2 of the Office Action, the drawings have been objected to for having failed to reference the "virtual machine 100" recited in the specification. In response, the Applicants have amended the specification to remove the reference to the label "100". In paragraph 3, the use of the trademark JAVA has been objected to for lacking capitalization in pertinent portions of the specification. In response, the Applicants have amended selected sections of the specification to distinguish between the trademark usage of the term JAVA and the non-trademark, generic usage of the term Java to refer to the Java programming language, much in the same way that C++ or COBOL refer to other program languages generically without regard to the source or origin in commerce of a compiler or runtime environment for the programming language.

Presently, claims 1 through 20 are pending in the Patent Application. Claims 1, 4 and 12 are independent claims. In paragraph 4 of the Office Action, claims 1-3 have been rejected under 35 U.S.C. §101 for having been directed to non-statutory subject matter by reciting merely a "custom class loader". In response and in accordance with the Examiner's helpful suggestion, the Applicants have amended the preamble of each of claims 1-3 to recite a "custom class loader apparatus". Notably, only the preamble and not the limitations of claims 1-3 have been amended to overcome the foregoing rejection under 35 U.S.C. §101.

In paragraphs 5 through 7, claims 3 and 17 have been rejected under 35 U.S.C. §112, second paragraph as being indefinite for failing to particularly point out and distinctly claim the

subject matter which the Applicants regard as the invention. Specifically, claim 3 recited the term JAVA indicating a trademark usage. Moreover, claim 17 recited "each parent class loader" which appears to lack antecedent basis. In response, the Applicants have amended claim 17 to recite "at least one" in lieu of "each" in order to obviate the rejection under 35 U.S.C. §112, second paragraph.

In respect to claim 3, the Applicants have amended claim 3 to recite the term "Java" as a non-trademark usage referring to the Java programming language and not as a source indicator for any particular Java compiler or runtime environment. In fact, the Java runtime environment, version 1.2 is known to be a general specification available for implementation by any number of manufacturers including Microsoft Corporation and IBM Corporation, for example. To that end, the term Java does not enjoy trademark protection when used generically to refer to a runtime environment which comports with a set of specifications for the Java programming language. Accordingly, the phrase Java version 1.2 is not a trademark usage and claim 3 has been amended accordingly.

In paragraphs 8 and 9 of the Office Action, claims 1-20 have been rejected under 35 U.S.C. §103(a) as being unpatentable (obvious) over United States Patent Application Publication No. 2004/0015936 to Susaria et al. ("Susaria"). In response, the Applicants respectfully traverse the rejections on the art as the Applicants believe that Susaria does not support a *prima facie* case of obviousness as required under the Patent Act and defined in the Manual of Patent Examining Procedure, section 2142. Prior to a more in depth discussion of the rejections on the art, however, a brief review of the Applicants' invention is appropriate.

The Applicants have invented a method, system and apparatus for a custom class loading method and system which can efficiently process cyclically dependent classes in a virtual machine environment. In accordance with the inventive arrangements, each class loader in the system can include a list of peer class loaders. The list of peer class loaders can include those peer class loaders which are to be visited according to a dependency specification of an associated application in the virtual machine environment. Prior to constructing each class loader, the peer class loader list can be generated by traversing each dependency referred to in the dependency specification.

Each class loader also can include a “dirty bit”. The dirty bit can indicate when a new class loader should be created by virtue of a newly replaced class which is associated with the class loader. When the dirty bit of a class loader has been set, each class loader in the class loader list also can have the dirty bit set. Additionally, when a class loader is encountered which has the dirty bit set, a new class loader can be created. Notably, the dirty bit permits the deferral of class loader construction in order to reduce runtime overhead.

Importantly, the class loader list, once generated, guarantees that there will be a well known class loader ordering despite the cyclic references of the inter-dependent classes. Furthermore, the peer arrangement of the class loader list frees the class loading order from the restrictive parent/child arrangement of the prior art. Hence, the combination of the peer class loader list and the dirty bit provide a resource efficient mechanism for performing class loading amongst cyclically dependent classes.

Turning now to the rejections on the art, Susaria relates to dynamic class reloading using a modular, pluggable and maintainable class loader. As stated in the Susaria patent application

specification, each application in an application server can include a dynamic class loader module. The class loader module can include a hierarchical stack of class loaders. Each module in the application can be associated with its own class loader and each class loader can be responsible for loading one or more classes. In Susaria, when a class is changed, the changed class can be detected by the class loader module. Any notification for a class change can come to the class loader controller so that the concerned class loader can be replaced. Moreover, the class loaders for all classes that depend on the changed class can be replaced. Subsequently, the replaced class loaders can reload the affected classes.

Referring first to claim 1 of the Patent Application, claim 1 as amended recites a custom class loader apparatus having the following limitations:

- (A) class loading logic configured to specifically and dynamically locate, define and load a class specified by name;
- (B) a list of peer class loaders arranged in accordance with the associated dependency specification and, list generation logic configured to generate said list when said specified class has been replaced or when said dependency specification has been modified;
- (C) a flag indicating whether said class has been replaced; and,
- (D) deference logic configured to defer said location, definition and loading of said specified class to said peer class loaders in said list.

Considering limitations B, C and D, paragraph 56 of the Susaria patent application specification fails to recite, teach or even suggest a "list of peer class loaders arranged in accordance with the associated dependency specification, and list generation logic configured to generate said list when said specified class has been replaced or when said dependency

specification has been modified". Rather, paragraph 56 of Susaria merely refers to "a hierarchical stack of class loaders". As for the recited "flag" element of limitation C, the "dirty class monitor" of paragraph 59 of the Susaria patent application cannot be equated to a "flag" when giving the term "flag" its ordinary and plain meaning. Finally, the "deference logic" of limitation also cannot be located anywhere within paragraph 59 of the Susaria specification. In fact, no where in Susaria is it ever suggested that any logic "defers" any operation to a "peer class loader" in a "list" as explicitly required by the plain language of claim 1.

Referring next to claims 4 and 12, a process of coordinating class loading among cyclically dependent classes is recited having the following limitations:

- (W) receiving a request to load a specified class;
- (X) determining whether said specified class has been replaced;
- (Y) if it is determined that said specified class has been replaced, constructing a new instance of the class loader and generating a list of peer class loaders to which location, definition and loading of said specified class are to be deferred in accordance with a dependency specification in the virtual machine; and,
- (Z) deferring said location, definition and loading to said peer class loaders in said list.

Again, no where in Susaria is it stated that a "list of peer class loaders" (as opposed to class loaders in of themselves) are generated to which "location, definition and loading" of a specified class is to be "deferred" in accordance with a dependency specification in a virtual machine. In fact, no "deferral" ever is described in Susaria. Rather, in Susaria, class loaders always remain associated with corresponding classes and class loaders can be replaced as

required when an associated class is changed. Accordingly, Susaria fails to teach each recited limitation of claims 4 and 12.

As required by Section 2142 of the Manual of Patent Examining Procedure (MPEP), to establish a prima facie case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art references when combined must reach or suggest all the claim limitations. Clearly, the recitation of Susaria cannot satisfy the three basic criteria of Section 2142.

Specifically, no evidence has been cited to indicate that a "flag" can be equated to a "dirty class monitor" which "monitor[s] classes used by the application and detect when any of the classes have been changed." Flags are passive and do not monitor other classes as suggested in the Office Action. Rather, flags are variables holding values. Aside from the "flag", no evidence has been cited to indicate that the "list of peer class loaders" can be equated to a "hierarchical stack of class loaders". Also, no evidence has been cited to indicate the presence of "list generation logic" in Susaria. Moreover, the fact that when a class has been replaced or modified, appropriate class loaders can be provided to dependent classes in Susaria bears no relation to "list generation logic". Finally, the act of "deferring" is never discussed in Susaria. Accordingly, Susaria alone cannot support a rejection under 35 U.S.C. § 103(a).

In sum, the Applicants believe that the originally filed claims 1-20 distinguish over the cited art and stand patentable and ready for an indication of allowance. As such, the Applicants respectfully request the withdrawal of the rejections under 35 U.S.C. §§ 101, 103(a) and 112

Application No. 10/026,266
Filed: 12/21/2001
Attorney Docket No.: RSW920010208US1

second paragraph based upon the foregoing remarks. This entire application is now believed to be in condition for allowance. Consequently, such action is respectfully requested. The Applicants request that the Examiner call the undersigned if clarification is needed on any matter within this Amendment, or if the Examiner believes a telephone interview would expedite the prosecution of the subject application to completion.

Respectfully submitted,

Date: December 27, 2004

Steven M. Greenberg
CUSTOMER NUMBER 46320
